Data Stripper Controller

Interface Control Document

for the

ADEOS II NASA/NOAA Ground Network (NGN)

December 10, 2000
Revision 14

Prepared by:
ViaSat Inc.
Communications and Tracking Systems
4311 Communications Drive
POB 1587
Mail Stop: ATL 30-J
Norcross, Georgia 30093

Table Of Contents

1.0 Document Overview

1.1 Overview of Important Changes

This document should be compared against Version 8. This document contains all the changes shown in Version 9 with the following important exceptions:

- Ephemeris has been eliminated.
- The L0RL file name follows the NASDA specification. (This is a change from Version 9.)
- The Asychronous status packet has been expanded.
- The TBDs have been eliminated.

1.2 Scope

This Interface Control Document (ICD) defines the interface specification for the ADEOS II NASA/NOAA Ground Network (NGN) Data Stripper Controller (DSC) and the following three external systems (subsystems).

- Remote host scheduling computer of the ADEOS II NASA/NOAA Ground Station Network (NGN) local site, either the Alaska SAR Facility (ASF) Host Controller or the Wallops Flight Facility (WFF) Master Computer. This remote interface provides the Master or Host the following functions:
  - Update of recording schedule (lists of satellite passes from which to extract data)
  - Reading of status logs (ASCII file showing Data Stripper processing results)
  - Reception of status messages showing the current state of the Data Stripper and the DSC itself.
- TSI Telsys, Inc. data stripper subsystem. This interface allows the DSC to control the operation of the data stripper, and access file and status information.
- SAFS file archival/distribution system. This interface allows the DSC to transfer files to the SAFS system for subsequent delivery to data customers.

2.0 Applicable Documents

NASDA ADEOS II Project Ground Segment Recording Sub-system Level 0 Format Descriptions for the following files:

- AMSR
- DCS
- DMS-1
- DMS-2
- TEDA
- GLI-1K
- Housekeeping (HK)

- ILAS
- SeaWinds
- VMS
- Format Description of Mission Operation Information Files (MOIF) between NASDA and NASA (AD2-EOSD-98-155)
- Data Stripper Documentation
- SAFS Documentation

3.0 System Overview

3.1 System diagram

Figure 2.1 shows the DSC in relation to other Ground Station components.

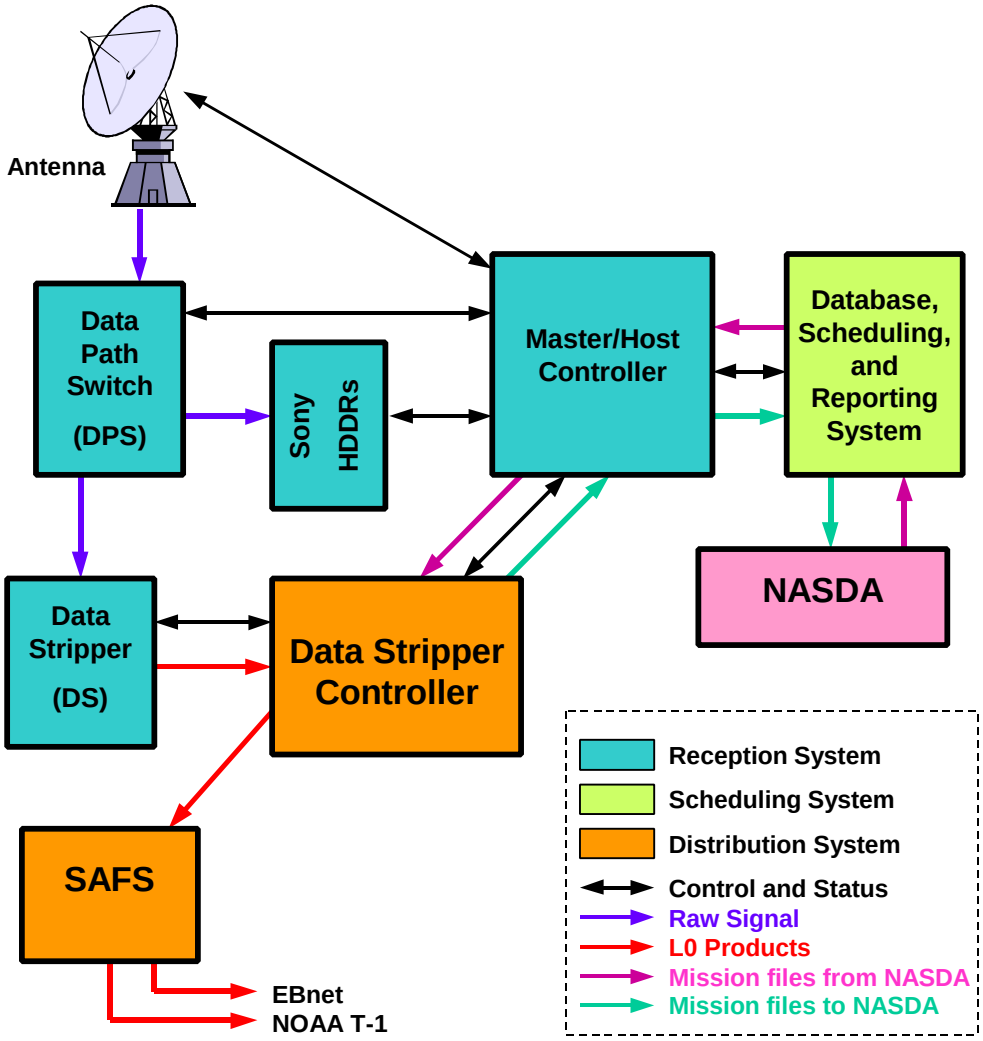


Figure 2.1 ASF/WFF Simplified Block Diagram

4.0 Interfaces

Figure 4.0 shows the information flow between the DSC and the other Ground Station components.

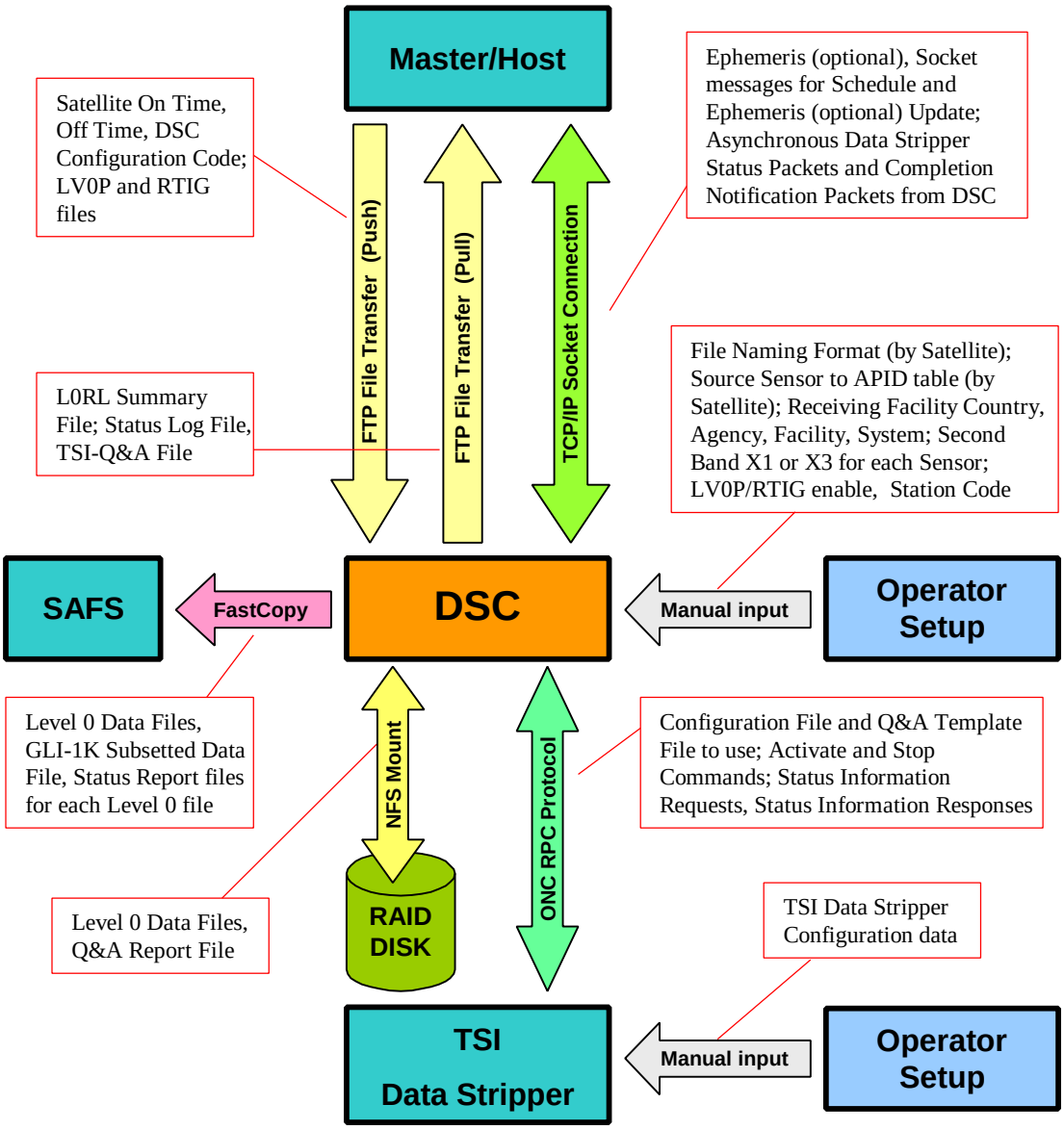


Figure 4.0 Data Stripper Software Data Flow

4.1 Between DSC and Master/Host Controller

4.1.1 Control, Status, and Completion Notification Packets

Socket messages are used between the DSC and Host/Master Controller for control and status, and to communicate completion of events. See Appendix A for more details.

4.1.1.1 Master/Host Controller to DSC

1) **Control Request Packets**
   a) Service Name – rci_recv
   b) Port Number/Protocol– 3015/tcp
   c) Timing/Response – Asynchronous messages sent to communicate control events.
   d) Error Handling – Errors should be logged and a reconnection should be attempted.
   e) Implementation Constraints – none.

4.1.1.2 DSC to Master/Host Controller

1) **Control Response Packets**
   a) Service Name – rci_send
   b) Port Number/Protocol – 3015/tcp
   c) Timing/Response – Synchronous response messages sent as replies to control messages.
   d) Error Handling – Errors should be logged and a reconnection should be attempted.
   e) Implementation Constraints – none.

2) **Asynchronous Status Packets**
   a) Service Name – rci_send
   b) Port Number/Protocol – 3015/tcp
   c) Timing/Response – Asynchronous status messages sent at interval requested.
   d) Error Handling – Errors should be logged and a reconnection should be attempted.
   e) Implementation Constraints – none.

3) **Asynchronous Completion Notification Packets**
   a) Service Name – rci_send
   b) Port Number/Protocol – 3015/tcp
   c) Timing/Response – Asynchronous messages sent at the completion of LZP processing and transfer of all files to the SAFS.
   d) Error Handling – Errors should be logged and a reconnection should be attempted.
   e) Implementation Constraints – none.

4.1.2 Files

4.1.2.1 Master/Host Controller to DSC

1) **SSP (Schedule and Setup Parameters)** [General]
   a) Format – ASCII text as shown in Appendix A.
   b) Timing/Response – File transfer to DSC must be completed before Schedule Update command is given.
   c) Protocol – FTP (push) from Master/Host Controller to DSC, followed by socket connect "Update Schedule" command. See Appendix A.
   d) Error Handling – On inability to complete FTP or socket connection command, notify Master/Host Controller console and log event.
   e) Implementation Constraints – The transfer must be of a full schedule (one or more supports) as described in Appendix A.. Incremental schedule changes are not permitted. Schedule should be a minimum of one support, and a maximum of one day's supports to make sure that the supports are included in previously sent LV0P and RTIG files.
   f) File Naming Convention – $ETCDIR\remote\rciSched

2) **LV0P and RTIG** [ADEOS-II specific]
   a) Format – ASCII text, fixed format, as shown in Appendices B and C, respectively.
   b) Timing/Response – Files are received 3 times per week from NASDA. Files will be forwarded immediately to Master/Host. Master/Host must FTP them to DSC prior to sending any schedule which includes them.
   c) Protocol – FTP (push) from Master/Host.
   d) Error Handling – On inability to complete FTP transfer to DSC, notify Master/Host Controller console and log event.
   e) Implementation Constraints – Files must be present at DSC prior to requesting a schedule update which includes passes included in the file data.
   f) File Naming Convention – $ETCDIR\remote\LV0Pnnnnnn. As stated in NASDA documentation, where nnnnnn is a unique monotonically increasing number.

4.1.2.2 DSC to Master/Host Controller

1) **DSC Status Log File**
   a) Format – ASCII text as shown in Appendix E.
   b) Timing/Response – Transfer of the file occurs upon Support (Pass) completion after DS shutdown completed.
   c) Protocol – FTP Master (pull) from DSC.
   d) Error Handling – On inability to complete FTP transfer, log error to Master/Host console.
   e) Implementation Constraints – During a DSC test a null file descriptor may be returned to the Master/Host Controller.
   f) File Naming Convention – The file name will be returned in the completion message sent from the DSC to the Remote Computer. The full file directory path listed in the message will be; $ETCDIR\log\reports\satname.ddd.hh.mm, where satname is obtained from the DSC configuration file. Example: "ADEOS2", ddd = 001 – 366, hh = 00 – 23, mm = 00 – 59.

2) **L0RL** [ADEOS-II specific]

NOTE: There will be 1 L0RL file for each downlink segment and a pass can consist of 1 or 2 downlink segments. Therefore, there will be 1 to 2 completion messages for the L0RL files.

a) Format – ASCII text as shown in Appendix D.
b) Timing/Response – Transfer of the file occurs upon Support (Pass) completion after file transfer to SAFS has been completed.
c) Protocol – FTP Master/Host (pull) from DSC.
d) Error Handling – On inability to complete FTP transfer, log error to Master/Host console.
e) Implementation Constraints – During a DSC test a null file descriptor may be returned to the Master/Host Controller.
f) File Naming Convention – The file name will be returned in the completion message sent from the DSC to the Remote Computer. The full file directory path listed in the message will be; $ETCDIR\remote\L0RLnnnnnn. As stated in NASDA documentation, where nnnnnn is a unique monotonically increasing number.

3) **Summary Q&A Report** File [General]
a) Format – ASCII as defined in the TSI documentation.
b) Timing/Response – Transfer of the file occurs upon Support (Pass) completion after DS shutdown completed.
c) Error Handling – On inability to complete FTP transfer, log error to Master/Host console.
d) Implementation Constraints - This file will not reside on the DSC. Its full file directory path will point to the Data Stripper. During a DSC test a null file descriptor may be returned to the Master/Host Controller in the completion message.
e) Naming Convention –The file name will be returned in the completion message sent from the DSC to the Remote Computer. The full file directory path listed in the message will be;

ftp://*domain name/path/*PRI_QA_REPORT.satname.ddd.hh.mm and ftp://*domain name/path*/SEC_QA_REPORT.satname.ddd.hh.mm, where satname is obtained from the DSC configuration file. Example: "ADEOS2", ddd = 001 – 366, hh = 00 – 23, mm = 00 – 59.

As an example:

ftp://dsc.sa.com/export/home/wallopds/qareports/PRI_QA_REPORT.ADEOS2.093.18.26

4.2 Between DSC and Data Stripper

4.2.1 Files

4.2.1.1 Data Stripper to DSC

1) **Data Files** [General]
   a) Format – Level 0 data files
   b) Timing/Response – All files will be available in priority order following LZP processing as described in the DS configuration file.
   c) Protocol – DS RAID is NFS-mounted to DSC. DSC will use Fastcopy to transfer files to SAFS.
   d) Error Handling – Will use services provided by Fastcopy.
   e) Implementation Constraints – none.
   f) File Naming – The DS level 0 files are stored in a single directory on the RAIDs connected to the Primary and Secondary Level Zero Processors. These files have the name "apid_port_yyMMddhhmmss", where: "apid" and "port" are the decimal APID and port numbers; "yymmdd" and "hhmmss" are the numerical date and time where yy = 00 – 99, MM = 01 – 12, dd = 01 – 31, hh = 00 – 23, mm = 00 – 59, ss = 00 – 59.

4.2.2 Status & Commands

4.2.2.1 Data Stripper to DSC

1) System Characteristics
   a) Format – Programmatic RPC calls as defined in the TSI GMS API.
   b) Timing/Response – The function will be exercised at DSC startup. An error status will be returned indicating success or failure.
   c) Protocol – ONC RPC protocol
   d) Error Handling – On error, display notable communications error on screen
   e) Implementation Constraints – The GMS must be running.
   f) API Functions Used:
      gms_systeminfo_1 – To get system characteristics.

2) Subsystem Details
   a) Format – Programmatic RPC calls as defined in the TSI GMS API.
   b) Timing/Response – The function will be exercised at DSC startup. An error status will be returned indicating success or failure.
   c) Protocol – ONC RPC protocol
   d) Error Handling – On error, display notable communications error on screen
   e) Implementation Constraints – none.
   f) API Functions Used:
      gms_getknownsubsystems_1 – To get subsystem information.

3) Standard Commands
   a) Format – Programmatic RPC calls as defined in the TSI GMS document API chapter and Monitoring and Controlling Gateway Systems chapter section 3.2.1.2.

b) Timing/Response – These functions will be exercised during prepass, pass, and postpass. An error status will be returned indicating success or failure.
c) Protocol – ONC RPC protocol
d) Error Handling – On error, display notable communications error on screen
e) Implementation Constraints – The GMS must be running. The usage order of these routines with others is important. See the Data Stripper API Implementation Constraints section for details.
f) API Functions Used:
gms_shutdown_1 – To indicate the end of the pass and to begin LZP processing. This function will automatically execute the gms_disable_1 and gms_flush_1.
gms_zerostatus_1 – Clears all telemetry quality and accounting statistics to zero.

4) Configuration Set Commands

a) Format – Programmatic RPC calls as defined in the TSI GMS document API chapter.
b) Timing/Response – These functions will be exercised at the beginning of each pass. An error status will be returned indicating success or failure.
c) Protocol – ONC RPC protocol
d) Error Handling – On error, display notable communications error on screen
e) Implementation Constraints – The GMS must be running. The usage order of these routines with others is important. See the Data Stripper API Implementation Constraints section for details.
f) API Functions Used:
gms_activatecs_1 – To activate the configuration set. This function will automatically execute the gms_reset_1, gms_loadcs_1, and gms_enable_1.

5) Status

a) Format – Programmatic RPC calls as defined in the TSI GMS API.
b) Timing/Response – These functions will be exercised during idle time, prepass, pass, and postpass. An error status will be returned indicating success or failure.
c) Protocol – ONC RPC protocol
d) Error Handling – On error, display notable communications error on screen
e) Implementation Constraints – The GMS must be running.
f) API Functions Used:
gms_taggroupstatus_1 – To tag a group of parameters for later retrieval.
gms_gettaggedstatus_1 – To get a group of tagged parameters.

6) Q&A Reports

a) Format – Programmatic RPC calls as defined in the TSI GMS API.
b) Timing/Response – This function will be exercised during idle time, prepass, pass, and postpass. An error status will be returned indicating success or failure.
c) Protocol – ONC RPC protocol
d) Error Handling – On error, display notable communications error on screen
e) Implementation Constraints – The GMS must be running.
f) API Functions Used:
gms_qarequest_1 – To generate a Q&A report

7) Unsolicited Messages

a) Format – Programmatic sockets as defined in the TSI GMS API, section 7.11 and 8.2.
b) Timing/Response – This function will be exercised during idle time, prepass, pass, and postpass. An error status will be returned indicating success or failure.
c) Protocol – Socket protocol
d) Error Handling – On error, display notable communications error on screen
e) Implementation Constraints – The GMS must be running. If the socket connection is lost the client (DSC) will attempt to reconnect.
f) API Control Message Structures:
System 0 Availability Message – All ASF and WFF systems are System 0.
Quality & Accounting Report Generated Message.
Quality & Accounting Report Generation Failed Message.
Session Initiated Message – Indicates the requested DS activity is executing.
Session Complete Message – Indicates the requested DS activity is finished.
Session terminated by User – Indicates the requested DS activity was canceled by the user.
Session terminated Due to Error – Indicates the requested DS activity did not complete successfully.
Session Start – Indicates the CGS detects the frame synchronization pattern.
Session End – Indicates the CGS detects the loss of the frame synchronization pattern.
Level Zero Processing Started – Indicates LZP processing has started. Should get one of these messages per DS activity.
Level Zero Processing File Created – Indicates an LZP file has been created. DSC will get one of these for each file.
Level Zero Processing Ended – Indicates LZP processing has completed. Should get one of these messages per DS activity.
Level Zero Processing Aborted – Indicates LZP processing was aborted and did not complete successfully. Should get one of these per DS activity.

4.2.3 Data Stripper API Implementation Constraints

Prior to a support the status of all processing will be cleared by the DSC using the gms_zerostatus_1 function.

The DSC will use the gms_activatecs_1 API function to set up the system for data processing. The gms_activatecs_1 API function executes gms_reset_1, gms_loadcs_1 and gms_enable_1 control commands. When all data has been received the DSC will use the gms_shutdown_1 API function to shutdown the system and begin LZP processing.

4.3 Between DSC and SAFS

4.3.1 Files

4.3.1.1 DSC to SAFS

1) **Data Files** [General] (Excludes GLI-1K data files)
   a) Format – Level 0 data files as generated by the TSI Data Stripper and residing on DS RAID.
   b) Timing/Response – Files will be transferred to the SAFS at the conclusion of LZP processing according to the priorities established on the DS.
   c) Protocol – FastCopy push by DSC from DS Raid to SAFS Raid
   d) Error Handling – Will use services provided by Fastcopy.
   e) Implementation Constraints – none.
   f) File Naming – Files will be renamed during FastCopy activity to meet SAFS specifications. See SAFS specification at http://www.wff.nasa.gov/~web/safs/. See Appendix F for level zero data file naming conventions.

2) **GLI-1K Data Files**
   a) Format – Level 0 GLI-1K data files as generated by the TSI Data Stripper and residing on DS RAID.
   b) Timing/Response – Files will be subsetted at the conclusion of LZP processing according to the priorities established for the DS/DSC/SAFS system.
   c) Protocol – File I/O from NFS mounted DS RAID to Subset Application then FastCopy push by DSC from DSC to SAFS.
   d) Error Handling – Will use services provided by File I/O and Fastcopy.
   e) Implementation Constraints – none.
   f) File Naming – Files will be renamed during subsetting activity to meet SAFS specifications. See SAFS specification at http://www.wff.nasa.gov/~web/safs/. See Appendix F for level zero data file naming conventions.

3) **Metadata (Status) Files** [General]
   a) Format – ASCII files generated by DSC using quality data from DS
   b) Timing/Response – Metadata files can only be generated at the conclusion of LZP processing and for GLI-1K at the conclusion of subsetting.
   c) Protocol – FastCopy push from DSC to SAFS Raid.
   d) Error Handling – Will use services provided by Fastcopy.
   e) Implementation Constraints – none.
   f) File Naming – See SAFS specification at http://www.wff.nasa.gov/~web/safs/. See Appendix F for level zero data file naming conventions.

APPENDIX A.

Detailed Protocol for Communications Between DSC and Remote Host Computer

5.0 Appendix A - Detailed Communications Protocol

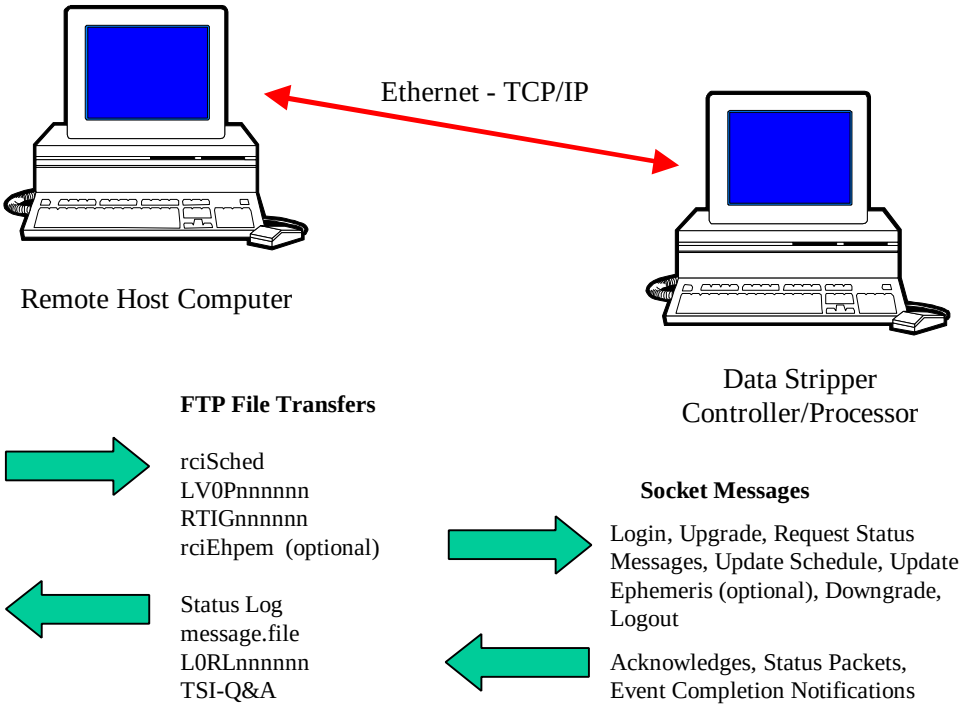
5.1 Introduction

5.1.1 Scope

This Appendix defines the detailed interface between the Data Stripper Controller/ Processor (DSC) and the remote host computer of the ADEOS II NASA/NOAA Ground Station Network (NGN) local site, either the Alaska SAR Facility (ASF) Host Controller or the Wallops Flight Facility (WFF) Master Computer. This remote interface allows for a remote computer to update schedules (lists of satellite passes from which to extract data), read status logs (ASCII file showing Data Stripper status), and receive status messages which show the current state of the Data Stripper and the DSC itself.

6.0 Appendix A - Interface Specification

6.1 Interface Diagram



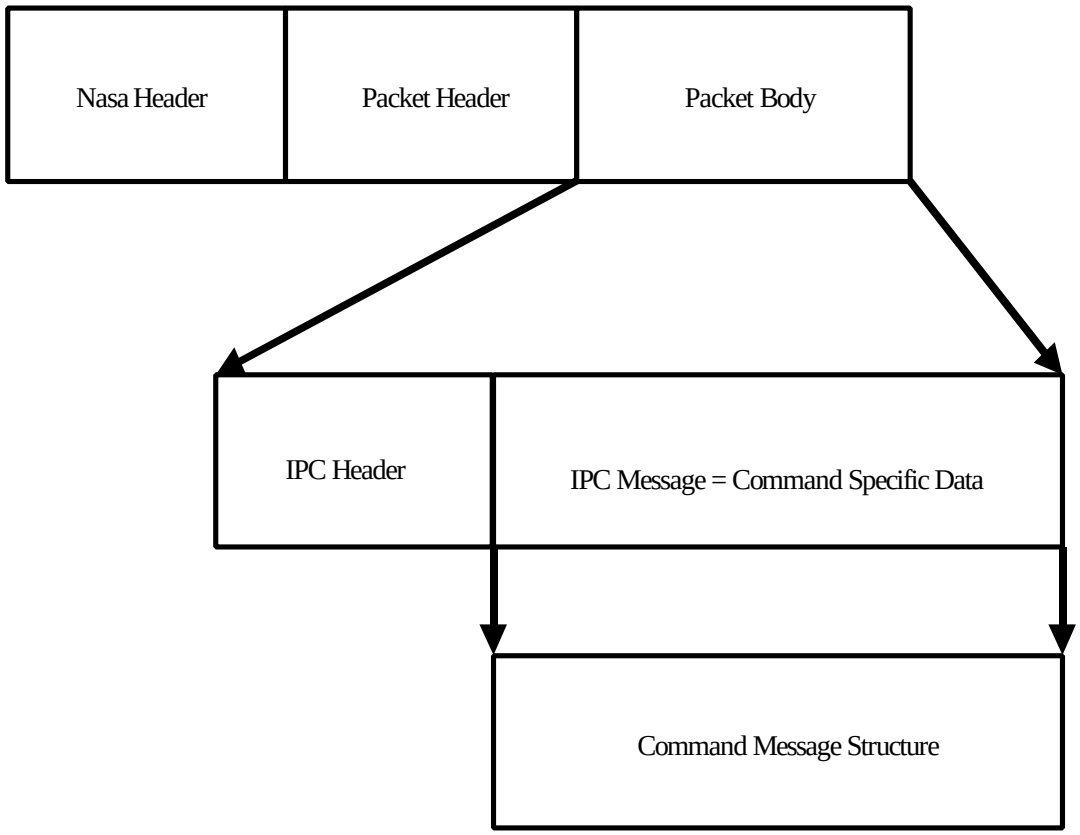
7.0 Appendix A - Interface Design

This interface has evolved from prior specifications requiring the minimum capability to download schedules, and upload pass logs. This interface has been proven in a number of installations. The message.file file allows a remote computer to have access to the messages that are displayed in either the status or error windows of the top level screen of the DSC.

Files are transferred to and from the DSC via FTP. In order to tell the DSC that there is new schedule to process, socket messages are used. These socket messages are described below.

7.1 TCP/IP Packet Format

Within the scope of this document, the term ***packet*** is construed to mean a construct composed of a standard NASA 272-byte header and data (described in section A3.1.3), followed by a DSC Remote Interface packet header structure and a packet body structure (see figure below). The size of the packet header is fixed, but the packet body is of variable length depending on the command message sent. The IPC header contains information necessary to support data routing requirements for both the DSC and the Remote Computer. The DSC sends and receives packets in Big Endian order (high byte of data sent first).



7.1.1 NASA Header

The data type NASA_HEADER, shown below, provides the template for the standard 272-byte NASA header and data field preceding the DSC header and data portions. The INT data type defines a 4 byte field.

```
/* struct NASA requires at beginning of each message */

#define NASA_STRING_LEN   16

typedef struct
{
   CHAR site_identifier[NASA_STRING_LEN];
            /* physical location of sender/recipient */
   CHAR system_name[NASA_STRING_LEN];
            /* software system which produced message */
   CHAR logical_unit_name[NASA_STRING_LEN];
            /* name of computer on which software is running */
   INT address_type;          /* 0 - socket; 1 -  windows NT pipe */
   CHAR box_id[(2*NASA_STRING_LEN)];      /* unique 'port' id */
   INT task_id;               /* process id of the task */
   INT spare1;
   INT spare2;
} NASA_ADDRESS;

typedef struct
{
   INT word_byte_ordering;    /* flag: 0 = MSB first; 1 = LSB first */
   INT extended_header_type;  /* extended header present: 0=no; 1=yes */
   INT body_size;             /* bytes used in message body */
   INT body_block_size;
            /* 0=no blocking; non-zero=block size in bytes */
   NASA_ADDRESS source_address;     /* identifies message sender */
   NASA_ADDRESS dest_address;       /* identifies message recipient */
   CHAR group[NASA_STRING_LEN];     /* group name */
   INT category;                    /* enum */
   INT type;                        /* enum */
   INT unit;
   INT ID;
   INT is_certified;
   INT priority;
   INT security_key;
   INT time_stamp_year;       /* year of message; 4 digits */
   INT time_stamp_day;        /* day of the year */
   INT time_stamp_seconds;    /* seconds past midnight */
   INT time_stamp_frac_sec;   /* fraction of seconds */
   INT spare;
} NASA_HEADER;
```

7.1.2 DSC Packet Header

The data type PACKET_HEADER is the template for a Remote Control Interface (RCI) packet header.

```
typedef struct {
      LONG ByteCount;
} PACKET_HEADER;
```

Field content is detailed in the following table:

Table 7-1: Remote Control Interface Packet Header Fields

Field Name	Bytes	Description
ByteCount	**4**	**Packet size in bytes, including header**

The ByteCount has a range of values from 4 to 128.

The address of the first byte of the packet body is given by the following C code fragment

```
PACKET_HEADER * pHeader;
void* pvPacketBody;

pvPacketBody = (char*)pHeader + sizeof(PACKET_HEADER);
```

7.1.3 DSC Packet Body

The packet body is defined by the interprocess communication (IPC) command message types The IPC message types are login, control upgrade, update schedule, control downgrade and logout and the associated responses. The IPC command message is a binary structure obeying the formats and conventions discussed below. An additional command message type is defined for the Remote Computer to request that system status packets be sent from the DSC on the socket connection. This status request message is described in section A3.A3.7.

7.2 Interprocess Communication Command Message Format

7.2.1 IPC Message Header

Table 7-2: Message Header Format

Field	Data Type	Value	Description
destPid	pid, 4 byte enum	Varies	Process ID of message consumer
message_id	msgid, 4 byte enum	Varies	Integer message ID
process_id	pid, 4 byte enum	Varies	Process ID of the sender
original_pid	pid, 4 byte enum	14 (Always!)	Process ID of response consumer
o_rsp_msgid	msgid, 4 byte enum	Varies	Message ID response consumer expects the response to assume
message_length	4 byte INT	Varies	Length of message in bytes

Note: For specific examples of packet header values see later sections of this Apprendix.

7.2.2 Remote Control Interface Message Ids

```
typedef enum {
      RCI_Connect             =  6,
      RCI_Disconnect          =  7,
      RCI_AuthResponse        =  8,
      RCI_EupResponse         =  9,
      RCI_SupResponse         = 10,
      RCI_NotInControl        = 11,
      RCI_StatusReport        = 12,
      RCI_CompletionNotice    = 13,
      last_rci_message_id     = 14
} rci_message_ids;
```

7.2.3 Identification of Responses

As shown above in Table 7-2, the field o_rsp_msgid provides the means by which the sender of a command message may designate the ID with which the response message is tagged. The exception case is when a Schedule Update request is sent to the DSC when the sender is not logged in. In that case, an Error Message is returned with a fixed Message ID of 11 (Not in Control). The message header format for responses are listed below:

Table 7-3: Response Message IPC Header and IPC Message Format

Field	Data Type	Value	Description
destPid	4 byte enum	0 (Always!)	Process ID of message consumer
message_id	4 byte enum	o_rsp_msgid of header received	Integer message ID. Always 11 if not logged in.
process_id	4 byte enum	destPid of header received	Process ID of the sender
original_pid	4 byte enum	14 (Always!)	Process ID of response consumer
o_rsp_msgid	4 byte enum	0 (Always!)	Message ID response consumer expects the response to assume
message_length	4 byte INT	28 (Always!)	Length of message in bytes
status	4 byte INT	Varies	Depends upon message sent

7.2.4 Routing of Messages and Responses

To illustrate the above, an example is included that illustrates all of the possible messages that can be sent across the remote interface.

7.2.4.1 Login Message

```
TEST_SERVER: Send a login msg to the remote server

Packet sent:
           NASA header:  (272 bytes of data)
             ByteCount: 92
         header.destPid: 3
      header.message_id: 0
      header.process_id: 13
    header.original_pid: 14
     header.o_rsp_msgid: 8
 header.message_length: 88
         user_name[32]: operator
        user_passwd[32]: *

Packet Received:
           NASA header:  (272 bytes of  data)
             ByteCount: 32
         header.destPid: 0
      header.message_id: 8
      header.process_id: 3
    header.original_pid: 14
     header.o_rsp_msgid: 0
 header.message_length: 28
                  data: 0  [status = auth_passed]
```

7.2.4.2 Control Upgrade Message

```
TEST_SERVER: Send a Control Upgrade msg to the remote server

Packet sent:
           NASA header:  (272 bytes of data)
             ByteCount: 92
        header.destPid: 3
     header.message_id: 2
     header.process_id: 13
   header.original_pid: 14
    header.o_rsp_msgid: 8
 header.message_length: 88
         user_name[32]: operator
       user_passwd[32]: *

Packet Received:
           NASA header:  (272 bytes of data)
             ByteCount: 32
        header.destPid: 0
     header.message_id: 8
     header.process_id: 3
   header.original_pid: 14
    header.o_rsp_msgid: 0
 header.message_length: 28
                 data: 5  [status = auth_grant]
```

7.2.4.3 Schedule Update Message

```
TEST_SERVER: Send a Schedule Update msg to the remote server

Packet sent:
           NASA header:  (272 bytes of data)
             ByteCount: 28
        header.destPid: 17
     header.message_id: 16
     header.process_id: 13
   header.original_pid: 14
    header.o_rsp_msgid: 10
 header.message_length: 24

Packet Received:
           NASA header:  (272 bytes of data)
             ByteCount: 32
        header.destPid: 0
     header.message_id: 10
     header.process_id: 17
   header.original_pid: 14
    header.o_rsp_msgid: 0
 header.message_length: 28
                 data: 1  [status = FILE_OPEN, error]
```

7.2.4.4 Control Downgrade Message

```
TEST_SERVER: Send a Control Downgrade msg to the remote server

Packet sent:
           NASA header:  (272 bytes of data)
             ByteCount: 92
        header.destPid: 3
     header.message_id: 3
     header.process_id: 13
   header.original_pid: 14
    header.o_rsp_msgid: 8
 header.message_length: 88
         user_name[32]: operator
       user_passwd[32]: *

Packet Received:
           NASA header:  (272 bytes of data)
             ByteCount: 32
        header.destPid: 0
     header.message_id: 8
     header.process_id: 3
   header.original_pid: 14
    header.o_rsp_msgid: 0
 header.message_length: 28
                 data: 8  [status = auth_downgrade]
```

7.2.4.5 Request Status Packets Message

```
TEST_SERVER: Send a Request Status Packets msg to the remote server

Packet sent:
           NASA header:  (272 bytes of data)
             ByteCount: 40
         header.destPid: 13
      header.message_id: 1
      header.process_id: 13
    header.original_pid: 14
     header.o_rsp_msgid: 8
 header.message_length: 36
           status modes: 0   [currently unused]
            update time: 10  [send status every 10 seconds]

Packet Received every 10 seconds (no normal acknowledge):

           NASA header:  (272 bytes of data)
             ByteCount: 28+sizeof(status_data)
         header.destPid: 0
      header.message_id: 12
      header.process_id: 14
    header.original_pid: 14
     header.o_rsp_msgid: 0
 header.message_length: 24+sizeof(status_data)
           status_data: See examples in Appendix A.
```

7.2.4.6 Completion Notification Message

```
Packet Received asynchronously at the end of LZP processing and file
transfer to SAFS

           NASA header:  (272 bytes of data)
             ByteCount: 28+sizeof(completion_data)
         header.destPid: 0
      header.message_id: 13
      header.process_id: 14
    header.original_pid: 14
     header.o_rsp_msgid: 0
 header.message_length: 24+sizeof(completion_data)
       completion_data: See examples in Appendix A.
```

7.2.4.7 Logout Message

```
TEST_SERVER: Send a Logout msg to the remote server

Packet sent:
           NASA header:  (272 bytes of data)
             ByteCount: 92
         header.destPid: 3
      header.message_id: 1
      header.process_id: 13
    header.original_pid: 14
     header.o_rsp_msgid: 8
 header.message_length: 88
          user_name[32]: operator
        user_passwd[32]: *

Packet Received:
           NASA header:  (272 bytes of data)
             ByteCount: 32
         header.destPid: 0
      header.message_id: 8
      header.process_id: 3
    header.original_pid: 14
     header.o_rsp_msgid: 0
 header.message_length: 28
                  data: 2  [status = auth_logout]
```

7.3 Command Message Structures

The following section describes the command set for the Remote Computer. The commands include:

Command	Possible Responses

Login - Used to gain access to the DSC.

- Login Response with auth_passed if okay, auth_failed if password does not verify for the user, auth_full if MAXIMUM number of simultaneous logins is exceeded.

Control Upgrade - Used to gain control access to the DSC.

- Control Privilege Response with auth_grant if request granted, auth_deny if user not authorized for control or not logged in.
- Control Privilege Response with auth_request if operator attempts to upgrade while Remote Computer is in control.

Schedule Update - Used to give new schedule data to the DSC.

Can contain one event or up to one week of events.

- Error message if not logged in.
- Schedule Update Response of SCHEDULE_UPDATED if successful,
- FILE_OPEN if unable to open the file,
- BAD_ACTIVITY if syntax error in an activity table.
- START_TABLE if missing start table delimiter, or
- END_TABLE if missing end table delimiter.
- Other error responses based on reading of the LV0P and RTIG files will indicate if the activities in the LV0P and RTIG did not match activities in the schedule.

AUT_ControlKeep - To keep control or no response within 30 seconds loses control.

- Control Privilege Response with auth_revoke if control revoked.

Control Downgrade - Used to relinquish control back to the operator.

- Control Downgrade Response with auth_downgrade status.

Logout - Used to end DSC access.

- Logout Response with auth_logout.

7.3.1 Login

The following defines the packet that the Remote Computer sends to the DSC to request a user login. A valid user name is a name that is registered within the PC's list of authorized logins. This is registered by system administration on the Windows/NT Operating System. A valid password is one that matches the first password used when an operator logged in to the DSC screen or when an operator first logged in remotely. There is no mechanism to change a password after the first use. However, there is a mechanism to eliminate memory of all the user's passwords, so that the next typed password becomes the new valid password. This is accomplished by erasing the file authentication.file in the etc directory.

A logged in user may request control upgrade.

7.3.1.1 Packet Header

Table 7-4: Login Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	Sizeof(PACKET_HEADER) + sizeof(aut_login_attempt_message)

7.3.1.2 Packet Body

The data type aut_login_attempt_message defines the DSC login message.

```
#define NAME_SZ   (31+1)
#define PASSWD_SZ (31+1)

typedef struct {
      message_header    header;
      CHAR              user_name[NAME_SZ];
      CHAR              user_passwd[PASSWD_SZ];
} aut_login_attempt_message;
```

Table 7-5: Login Packet Body

Field Name	Derived Type	Bytes	Enum Value	Assigned Value
header.destPid	pid	4	3	authentication
header.message_id	msgid	4	0	AUT_LoginAttempt
header.process_id	pid	4	13	RCI_SERVER
header.original_pid	pid	4	14	RCI_CLIENT
header.o_rsp_msgid	msgid	4	8	RCI_AuthResponse
header.message_length		4		sizeof(aut_login_attempt_message)
user_name		32		"user name"
user_passwd		32		"user password"

7.3.1.3 Login Response

The following defines the packet the DSC sends to the Remote Computer in response to a login request.

7.3.1.4 Packet Header

Table 7-6: Login Response Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(auth_response_message_type)

7.3.1.5 Packet Body

```
typedef struct{
      message_header    header;
      auth_status       status;
} auth_response_messsage_type;

typedef  enum{
      /* response statuses for login */
      auth_passed,
      auth_failed,
      auth_logout,
      auth_busy,
      auth_full,

      /* response statuses for control request */
      auth_grant,
      auth_deny,
      auth_request,
      auth_downgrade,
      auth_revoke
} auth_status;
```

See Table 7-3 for the response message IPC header IPC message format

7.3.2 Logout

The following defines the packet that the Remote Computer sends to the DSC to logout. A successful logout terminates access between the Remote Computer and the DSC. If a user successfully logs in and never logs out, then a subsequent login may be denied because of too many simultaneous users.

7.3.2.1 Packet Header

Table 7-7: Logout Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	Sizeof(PACKET_HEADER) + sizeof(aut_logout_message)

7.3.2.2 Packet Body

```
typedef struct {
      message_header    header;
      CHAR              user_name[NAME_SZ];
      CHAR              user_passwd[PASSWD_SZ];
} aut_logout_message;
```

Table 7-8: Logout Packet Header

Field Name	Derived Type	Bytes	Enum Value	Assigned Value
header.destPid	pid	4	3	authentication
header.message_id	msgid	4	1	AUT_Logout
header.process_id	pid	4	13	RCI_SERVER
header.original_pid	pid	4	14	RCI_CLIENT
header.o_rsp_msgid	msgid	4	8	RCI_AuthResponse
header.message_length		4		sizeof(aut_logout_message)
user_name		32		“user name”
user_passwd		32		“user password”

7.3.2.3 Logout Response

The following defines the packet the DSC sends to the Remote Computer in response to a logout request.

7.3.2.4 Packet Header

Table 7-9: Logout Response Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(auth_response_message_type)

7.3.2.5 Packet Body

```
typedef struct{
      message_header    header;
      auth_status       status;
} auth_response_messsage_type;

typedef  enum{
      /* response statuses for login */
      auth_passed,
      auth_failed,
      auth_logout,
      auth_busy,
      auth_full,

      /* response statuses for control request */
      auth_grant,
      auth_deny,
      auth_request,
      auth_downgrade,
      auth_revoke
} auth_status;
```

See Table 7-3 for the response message IPC header IPC message format

7.3.3 Control Upgrade Request

The following defines the packet that the Remote Computer sends to the DSC to request control access (upgrade). Granting control access to the Remote Computer allows the Remote Computer to update schedule.

7.3.3.1 Packet Header

Table 7-10: Control Upgrade Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(aut_control_request_message)

7.3.3.2 Packet Body

```
typedef struct {
      message_header    header;
      CHAR              user_name[NAME_SZ];
      CHAR              user_passwd[PASSWD_SZ];
} aut_control_request_message;
```

Table 7-11: Control Upgrade Packet Body

Field Name	Derived Type	Bytes	Enum Value	Assigned Value
header.destPid	pid	4	3	authentication
header.message_id	msgid	4	2	AUT_ControlUpgrade
header.process_id	pid	4	13	RCI_SERVER
header.original_pid	pid	4	14	RCI_CLIENT
header.o_rsp_msgid	msgid	4	8	RCI_AuthResponse
header.message_length		4		sizeof(aut_control_request_message)
user_name		32		“user name”
user_passwd		32		“user password”

7.3.3.3 Control Privilege Response

The following defines the packet the DSC sends to the Remote Computer in response to a control upgrade request.

7.3.3.4 Packet Header

Table 7-12: Control Upgrade Response Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(auth_response_message_type)

See Table 7-3 for the response message IPC header IPC message format

7.3.3.5 Packet Body

```
typedef struct{
      message_header    header;
      auth_status       status;
} auth_response_messsage_type;

typedef  enum{
      /* response statuses for login */
      auth_passed,
      auth_failed,
      auth_logout,
      auth_busy,
      auth_full,

      /* response statuses for control request */
      auth_grant,
      auth_deny,
      auth_request,
      auth_downgrade,
      auth_revoke
} auth_status;
```

7.3.4 Control Downgrade Request

The following defines the packet that the Remote Computer sends to the DSC to request control downgrade. After control downgrade, the local user (i.e. user interface) may upgrade to control access immediately without a thirty second delay.

7.3.4.1 Packet Header

Table 7-13: Control Downgrade Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(aut_control_request_message)

7.3.4.2 Packet Body

```
typedef struct {
      message_header    header;
      CHAR              user_name[NAME_SZ];
      CHAR              user_passwd[PASSWD_SZ];
} aut_control_request_messa ge;
```

Table 7-14: Control Downgrade Packet Body

Field Name	Derived Type	Bytes	Enum Value	Assigned Value
header.destPid	pid	4	3	authentication
header.message_id	msgid	4	3	AUT_ControlDowngrade
header.process_id	pid	4	13	RCI_SERVER
header.original_pid	pid	4	14	RCI_CLIENT
header.o_rsp_msgid	msgid	4	8	RCI_AuthResponse
header.message_length		4		sizeof(aut_control_request_message)
user_name		32		“user name”
user_passwd		32		“user password”

7.3.4.3 Control Downgrade Response

The following defines the packet the DSC sends to the Remote Computer in response to a control downgrade request.

7.3.4.4 Packet Header

Table 7-15: Logout Response Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(aut_control_request_message)

7.3.4.5 Packet Body

```
typedef struct{
      message_header    header;
      auth_status       status;
} auth_response_messsage_type;

typedef  enum{
      /* response statuses for login */
      auth_passed,
      auth_failed,
      auth_logout,
      auth_busy,
      auth_full,

      /* response statuses for control request */
      auth_grant,
      auth_deny,
      auth_request,
      auth_downgrade,
      auth_revoke
} auth_status;
```

See Table 7-3 for the response message IPC header IPC message format

7.3.5 Schedule Update

The following defines the packet that the Remote Computer sends to the DSC to load a remote schedule. A successful schedule update results in the current schedule being deleted and replaced with the new schedule. If performed during a support event, the event may be aborted. It will be restarted if it is included in the new schedule. The appropriate LV0P and RTIG files are read and the matching information extracted for inclusion in the event schedule. Please see section A3.4.1 for a sample remote schedule file.

7.3.5.1 Packet Header

Table 7-16: Schedule Update Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	Sizeof(PACKET_HEADER) + sizeof(header)

7.3.5.2 Packet Body

Table 7-17: Schedule Update Packet Body

Field Name	Derived Type	Bytes	Member Value	Assigned Value
header.destPid	pid	4	17	schedule_update
header.message_id	msgid	4	16	SUP_AddRemoteSchedule
header.process_id	pid	4	13	RCI_SERVER
header.original_pid	pid	4	14	RCI_CLIENT
header.o_rsp_msgid	msgid	4	10	RCI_SupResponse
header.message_length		4		sizeof(header)

7.3.5.3 Remote Schedule Update Response

The following defines the packet the DSC sends to the Remote Computer in response to a schedule update request. Note that this response may be delayed from the schedule update request message by many seconds, depending on the number of passes scheduled, as the DSC must read the schedule file and read the appropriate LV0P and RTIG files and correlate the event entries to build and validate the desired schedule.

7.3.5.4 Packet Header

Table 7-18: Schedule Update Response Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(sup_response_message_type)

7.3.5.5 Packet Body

```
typedef struct{
      message_header    header;
      SUP_STATUS        status;
} sup_response_messsage_type;

typedef  enum {
   SCHEDULE_UPDATED,    /* schedule successfully updated */
   FILE_OPEN,           /* error, couldn't open file */
   BAD_ACTIVITY,        /* syntax error, activity table */
   START_TABLE,         /* syntax error, no start table delimiter */
   SYNTAX_ERROR,        /* syntax error */
   MISSING_LV0P_FILE,   /* processing error, LV0P file missing */
   LV0P_ERROR,          /* data correlation error between LV0P and
                           schedule. In other words, LV0P or schedule
                           contains an activity that is not in the other
                           file */
   MISSING_RTIG_FILE,   /* processing error, RTIG file missing */
   RTIG_ERROR,          /* data correlation error between RTIG and
                           schedule. In other words, RTIG or schedule
                           contains an activity that is not in the other
                           file */
} SUP_STATUS;
```

See Table 7-3 for the response message IPC header IPC message format

7.3.6 Status Packet Request

The following defines the packet that the Remote Computer sends to the DSC to request asynchronous status packets. The status packets contain information as to the current state of the DSC and the Data Stripper. This request may disable the sending of the status packets, or enable them with a periodic rate of from 5 to 30 seconds between status updates. When the Remote Computer initially makes the connection to the DSC, the transmission of status packets is disabled.

7.3.6.1 Packet Header

Table 7-19: Asychronous Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(rci_status_request_message)

7.3.6.2 Packet Body

```
typedef struct {
      message_header    header;
      INT   mode;   /* status mode requested – unused */
      INT   time;   /* period in seconds: 0 or 5-30   */
} rci_status_request_message;
```

Table 7-20: Status Request Packet Body

Field Name	Derived Type	Bytes	Member Value	Assigned Value
Header.destPid	pid	4	13	RCI_SERVER
Header.message_id	msgid	4	12	RCI_SendStatusPackets
Header.process_id	pid	4	13	RCI_SERVER
Header.original_pid	pid	4	14	RCI_CLIENT
Header.o_rsp_msgid	msgid	4	12	RCI_SendStatusPackets
Header.message_length		4		sizeof(rci_status_request_message)
mode	INT	4	0	(unused)
time	INT	4	10	“send packets every 10 seconds”

7.3.6.3 Asynchronouos Status Report Packet Format

The following defines the packet the DSC sends to the Remote Computer to provide asynchronous status as requested by the status request message. This is not a response to the request, but an asynchronous data transmission occurring repeatedly at a time period specified by the status request

7.3.6.4 Packet Header

Table 7-21: Status Response Packet Header

Field Name	Base Type	Enum Value	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(status_report_message_type)

7.3.6.5 Packet Body

```
#define DSC_STATUS_SIZE (771)

typedef char DSC_status[DSC_STATUS_SIZE];

typedef struct{
      message_header    header;
      DSC_status        status;
} rci_status__message_type;
```

Table 7-22: Status Report IPC Header and IPC Message Format

Field	Data Type	Value	Description
destPid	4 byte enum	0 (Always!)	Process ID of message consumer
message_id	4 byte enum	o_rsp_msgid of header received	Integer message ID
process_id	4 byte enum	destPid of header received	Process ID of the sender
original_pid	4 byte enum	14 (Always!)	Process ID of response consumer
o_rsp_msgid	4 byte enum	0 (Always!)	Message ID response consumer expects the response to assume
message_length	4 byte INT	Varies	Length of message in bytes, maximum is 771
status	771 byte char	Varies	Depends upon message sent. See below.

Table 7-23: Determination of DSC_STATUS_SIZE

Keyword	K&V Delimiter	Value	K&V Pair Delimiter	Sub-string Length
TIME	=	1999-039T12:34:54.000	\n	27
SAFS_STATUS	=	" UNDEFINED "	\n	24
DSC_STATUS	=	" UNDEFINED "	\n	23
DSC_STATE	=	" AFTER_PREPASS "	\n	26
DSC_TEST_MODE	=	“ NOT_TESTING “	\n	28
P75_STATUS	=	" UNDEFINED "	\n	23
P75_STATE	=	" UNDEFINED "	\n	22
P75_SEARCH_ENTRIES	=	1234567890	\n	30
P75_BER	=	1.234E-090	\n	19
P75_RS_FRAMES	=	1234567890	\n	25
P75_RS_UC_ERRORS	=	1234567890	\n	28
P75_RS_CORR_ERRORS	=	1234567890	\n	30
P75_FIRST_LOCK	=	1999-039T12:34:54.000	\n	37
P10_STATUS	=	" UNDEFINED "	\n	23
P10_STATE	=	" UNDEFINED "	\n	22
P10_SEARCH_ENTRIES	=	1234567890	\n	30
P10_BER	=	1.234E-090	\n	19
P10_RS_FRAMES	=	1234567890	\n	25
P10_RS_UC_ERRORS	=	1234567890	\n	28
P10_RS_CORR_ERRORS	=	1234567890	\n	30
P10_FIRST_LOCK	=	1999-039T12:34:54.000	\n	37
S75_STATUS	=	" UNDEFINED "	\n	23
S75_STATE	=	" UNDEFINED "	\n	22
S75_SEARCH_ENTRIES	=	1234567890	\n	30
S75_BER	=	1.234E-090	\n	19
S75_RS_FRAMES	=	1234567890	\n	25
S75_RS_UC_ERRORS	=	1234567890	\n	28
75_RS_CORR_ERRORS	=	1234567890	\n	30
S75_FIRST_LOCK	=	1999-039T12:34:54.000	\n	37
End of string character			\0	1
Maximum length of DSC_STATUS message				771

Table 7-24: Values for DSC Status message.

Status values for SAFS, DSC , P75, P10, S75	Meaning
GOOD	For P75, P10, and S75: Software is AOK, status of HW is undetermined. For SAFS: Its heartbeat was received recently. For DSC: No anomaly has occurred during the pass.
BAD	For P75, P10, and S75: Software is reporting a problem, status of HW is undetermined. For SAFS: Its heartbeat has not been received recently. For DSC: An anomaly has occurred during the pass.
UNDEFINED	Transiant value, should be ignored.

Mode value for DSC test mode	Meaning
NOT_TESTING	DSC is not performing a test.
TESTING	DSC is performing a test pass.

State values for P75, P10, and S75 stream state	Meaning
LOCK	FEP is in lock mode.
FLYWHEEL	FEP is in flywheel mode.
SEARCH	FEP is in search mode.
CHECK	FEP is in check mode.
UNDEFINED	Transiant value, should be ignored.

Value of First Lock	Meaning
1970-000T00:00:00.000	The value 1970-000T00:00:00.000 is an initialization value that will be set at the beginning of program execution and at prepass and only updated if lock is detected.

Statue values for DSC state	Meaning
IDLE	No DSC activity is occurring.
PREPASS	DSC is in prepass initializing the strippers and cleaning up from the previous pass or test if either is still underway.
AFTER_PREPASS	DSC is idle awaiting pass start.
PASS	DSC is in pass, stripper is stripping data.
AFTER_PASS	DSC is shutting down strippers and LZP is underway.
AFTER_LZP	DSC is performing subsetting and transferring of data to SAFS.

Example of a ***DSC status*** variable:

TIME=1999-093T18:05:51.000\n
SAFS_STATUS=”GOOD”\n
DSC_STATUS=”GOOD”\n
DSC_STATE=” AFTER_PREPASS”\n
DSC_TEST_MODE=” NOT_TESTING”\n
P75_STATUS="GOOD"\n
P75_STATE="FLYWHEEL"\n
P75_SEARCH_ENTRIES=1234567890\n
P75_BER=1.234E-090\n
P75_RS_FRAMES=1234567890\n
P75_RS_UC_ERRORS=1234567890\n
P75_RS_CORR_ERRORS=1234567890\n
P75_FIRST_LOCK=1999-039T12:34:54.000\n
P10_STATUS="GOOD"\n
P10_STATE="FLYWHEEL"\n
P10_SEARCH_ENTRIES=1234567890\n
P10_BER=1.234E-090\n
P10_RS_FRAMES=1234567890\n
P10_RS_UC_ERRORS=1234567890\n
P10_RS_CORR_ERRORS=1234567890\n
P10_FIRST_LOCK=1999-039T12:34:54.000\n
S75_STATUS="GOOD"\n
S75_STATE="FLYWHEEL"\n
S75_SEARCH_ENTRIES=1234567890\n
S75_BER=1.234E-090\n
S75_RS_FRAMES=1234567890\n
S75_RS_UC_ERRORS=1234567890\n
S75_RS_CORR_ERRORS=1234567890\n
S75_FIRST_LOCK=1999-039T12:34:54.000\n\0

7.3.7 Asynchronous Completion Notification Message

The following defines the packet the DSC sends to the Remote Computer when it initially completes the support's data capture (Record state), and also when all files have been post-processed and sent to SAFS.

7.3.7.1 Packet Header

Table 7-25: DSC Completion Notification Packet Header

Field Name	Base Type	Enum Value	Assigned Value
ByteCount	LONG	4	Sizeof(PACKET_HEADER) + sizeof (completion_report_message_type)

7.3.7.2 Packet Body

```
#define DSC_COMPLETION_SIZE (329)

typedef char DSC_completion[DSC_COMPLETION_SIZE];

typedef struct{
      message_header    header;
      DSC_completion    complete;
} rci_completion_message_type;
```

Table 7-26: Completion Notification Message IPC Header and IPC Message Format

Field	Data Type	Value	Description
destPid	4 byte enum	0 (Always!)	Process ID of message consumer
message_id	4 byte enum	o_rsp_msgid of header received	Integer message ID.
process_id	4 byte enum	destPid of header received	Process ID of the sender
original_pid	4 byte enum	14 (Always!)	Process ID of response consumer
o_rsp_msgid	4 byte enum	0 (Always!)	Message ID response consumer expects the response to assume
message_length	4 byte INT	Varies	Length of message in bytes, maximum is 329
complete	329 byte char	Varies	Depends upon message sent. See below.

Table 7-27: Determination of DSC_STATUS_SIZE

Keyword	K&V Delimiter	Value	Delimiter	Sub-string Length
TIME	=	1999-039T12:34:54.000	\n	27
MSG_TYPE	=	" FILE_TRANSFER_FINISHED "	\n	34
FILE_PATH	=	" 254 chars for machine name, dir-path, & filename "	\n	267
End of string character			\0	1
Maximum length of DSC_COMPLETION variable				329

Note: The keyword (MSG_TYPE) can have a value of:

- LZP_FINISHED, *(This type of message will return a DSC Status Log File)*
- PRIMARY_Q&A_REPORT, *(This type of message will return a Q&A report)*
- SECONDARY_Q&A_REPORT, or
- FILE_TRANSFER_FINISHED *(This type of message will return a L0RL file)*

Examples of the value for the ***complete*** variable:

The ftp server on the DSC will have its base directory set to the "e:" drive. Therefore, the completion messages with a MSG_TYPE value of LZP_FINISHED or FILE_TRANSFER_FINISHED should have the first part of the string that matches, "ftp://strpboss/e:/", removed from the filepath when performing the ftp to fetch the files.

TIME=1999-093T18:05:51.000\n
MSG_TYPE="LZP_FINISHED"\n

FILE_PATH="ftp://strpboss/e:/DataStripper/curdev/bes/etc/reports/ADEOS2.093.18.05"\n\0

TIME=1999-093T18:05:51.000\n
MSG_TYPE="PRIMARY_Q&A_REPORT"\n
FILE_PATH="ftp://dsgems1/export/home/wallopds/qareports/PRI_QA_REPORT.ADEOS2.093.18.05"\n\0

TIME=1999-093T18:05:51.000\n
MSG_TYPE="SECONDARY_Q&A_REPORT"\n
FILE_PATH="ftp://dsgems2/export/home/wallopds/qareports/SEC_QA_REPORT.ADEOS2.093.18.05"\n\0

TIME=1999-093T18:05:51.000\n
MSG_TYPE="FILE_TRANSFER_FINISHED"\n
FILE_PATH="ftp://strpboss/e:/DataStripper/curdev/bes/etc/l0rl/L0RLnnnnnn.ADEOS2.093.18.05"\n\0

Note: During a DSC test a FILE_PATH with a value of null will be returned.

7.3.8 Not Logged In Message

The following defines the packet the DSC sends to the Remote Computer when it attempts to send a command prior to logging in.

7.3.8.1 Packet Header

Table 7-28: Error Message Packet Header

Field Name	Base Type	Bytes	Assigned Value
ByteCount	LONG	4	sizeof(PACKET_HEADER) + sizeof(rci_cmd_response)

7.3.8.2 Packet Body

```
typedef struct {
      message_header    header;
      message_ack       response;
} rci_cmd_response;

typedef  enum {
      ack = 1,
      nack
} message_ack;
```

NOTE: It is the field header.msgid with a value of RCI_NotInControl that determines the Remote Computer is not logged in. The value of response is always set to ack.

7.4 DSC System File Access

The Remote Computer will utilize standard FTP to send remote schedule files to the DSC and access the system status file. The remote host will provide the following files:

ftp://*dsc_hostname*/*$ETCDIR*/remote/rciSched
ftp://*dsc_hostname*/*$ETCDIR*/remote/LV0Pnnnnnn
ftp://*dsc_hostname*/*$ETCDIR*/remote/RTIGnnnnnn

The following files will be available to the remote computer:
http://*dsc_hostname*/*$ETCDIR*/message.file
http://*dsc_hostname*/*$ETCDIR*/logs/support/satname.ddd.mm.hh, satname.ddd.hh.mm, where satname is obtained from the DSC configuration file. Example: "ADEOS2", ddd = 001 – 366, hh = 00 – 23, mm = 00 – 59.

Note: The values of *$ETCDIR* and *dsc_hostname* will be defined when the system is installed.

Note: No notification message is sent to the remote computer regarding the file named message.file or the summary files. However, they can be retrieved at anytime using ftp.

7.4.1 Remote Schedule File- rciSched

The schedule file consists of one or more complete tables. One remote schedule file can schedule multiple events by defining a separate table for each event. The remote schedule must be submitted a nominal 5 minutes prior to track. The remote schedule replaces the current DSC schedule. Submitting an empty schedule file will delete the current schedule.

A table is divided into two components:

1. A table header block.
2. Activity description block

Each of these is discussed in the subparagraphs below.

Comments may be included by providing a '#' in column 1 of any line. Comments are skipped in the parsing of the table.

```
** OTS_SCH
*@ END OF HEADER
#
ACT_TYPE          NORMAL;
SCID              satname;
DSSID             33;
CONFIGURATION     name;
PROJ_PASS         12345;
ACT_NAME          VSOP231130500;
START_YEAR        2000;
START             231 13:05:00;
BOT               231 13:05:00;
EOT               231 23:00:00;
RISE_YEAR         2000;
RISE              231 12:55:00;
SET               231 23:30:00;
#
*= END =*
```

7.4.1.1 Header Block

The header block begins with a string that contains "** OTS_SCH". Note that there is a space between "**" and "OTS_SCH". The header block terminates with the string "*@ END OF HEADER"

```
** OTS_SCH
*@ END OF HEADER
```

7.4.1.2 Activity Schedule Block

```
ACT_TYPE          NORMAL;
SCID              satname;
DSSID             33;
CONFIGURATION     name;
PROJ_PASS         12345;
ACT_NAME          VSOP231130500;
START_YEAR        2000;
START             231 13:05:00;
BOT               231 13:05:00;
EOT               231 23:00:00;
RISE_YEAR         2000;
RISE              231 12:55:00;
SET               231 23:30:00;
```

ACT_TYPE specifies the activity type. NORMAL is the only valid tracking activity type.

SCID is the text satellite identifier. Where ***satnam****e* can only contain a maximum of 15 characters. Typically *satname* should match the satellite name specified in the DSC configuration file. For ADEOS2 the satellite name must have the first 5 characters as "ADEOS".

Examples:

ADEOS2 or ADEOS_TEST

DSSID is an integer tag up to 8 digits which is preserved in the scheduling data structure. Any 8 digits from 0 to 99999999 may be entered. Not applicable to MRST. This line is required.

CONFIGURATION is the configuration ID identifying the configuration to be used for the pass. Where ***name*** can only contain a maximum of 79 characters. The configuration ID is a string of the following form:

Normal Pass: alpha-numeric string with a .cfg file extension, or an ascii number such as: **adeos2_nominal.cfg** or **70**, and **adeos2_alternate.cfg** or **71**.

Test Pass: alpha-numeric string with a .tps file extension, such as: **adeos2_nominal_test.tps,** or **adeos2_alternate_test.tps**

PROJ_PASS is the orbit number.

ACT_NAME is an arbitrary string that can be used to tag the activity. Any arbitrary non-space string up to 14 characters may be entered.

START_YEAR is the year in which the scheduled activity starts. It must be in four digit format.

START is the start of the pass. This is not used by the scheduler, and is normally set to be the same as BOT. It is in the form <day of year><white space> <hh:mm:ss>. All times are UTC.

BOT is the Beginning of Track. This is what is used to schedule the activity. It is in the form <day of year><white space> <hh:mm:ss>. BOT is the time that program track begins. It differs from RISE in that RISE is the time when the satellite crosses the horizon. BOT is greater than or equal to RISE. All times are UTC.

EOT is the End of Track. This is what is used to determine the end time of the scheduled activity. It is in the form <day of year><white space> <hh:mm:ss>.

RISE_YEAR is the year in four digit format in which the satellite rises. It is normally the same as START_YEAR. The RISE_YEAR can be equal to or sooner than START_YEAR. It will be the same as START_YEAR unless the satellite crosses the horizon near the end of the day on the last day of a year and the BOT is after the RISE at the beginning of the first day of the next year.

RISE is the time the satellite is rising over the horizon. RISE is equal to or sooner than BOT. It only differs from BOT when it is desired to track at some time after it has crossed the horizon rather than as soon as it crosses the horizon. It is in the form <day of year><white space> <hh:mm:ss>.

SET is time the satellite is setting over the horizon. SET is equal to or later than EOT. It is in the form <day of year><white space> <hh:mm:ss>. It differs from EOT when it is desirable to stop tracking the satellite prior to it crossing the horizon rather than exactly when it crosses the horizon and becomes untrackable.

7.4.1.3 End of Table

```
#
*= END =*
```

The string *= END =* must be provided exactly in order to delimit end of table. Notice the single spaces leading and trailing the “END” string. Multiple schedules for multiple satellites may be specified in a single file. It is formatted by following the End of Table statement with another Table Header Block, Activity Block and End of Table, etc. There is an End statement per activity. Reaching end of file before finding another header block tells the software that the parsing is complete.

7.4.2 System Event Message File - message.file

The system event log contains the last 1000 error and status messages displayed on the DSC user interface. These messages are intended for debug purposes or to analyze errors and are subject to change.

The format specification is as follows:
The word ERROR, DEBUG or STATUS followed by two spaces if ERROR or DEBUG and one space if STATUS, followed by a task mnemonic in parenthesis (3-5 characters) followed by a space, a day and time stamp as shown below and a unique string. Please note that DEBUG Message strings all start with a number in parenthesis and ERROR and STATUS messages do not.. The format of the time is UTC. The following are some sample messages from the message.file:.

ERROR (ETSR) May 20 19:24:36:51 Socket signal not found.
ERROR (RFCL) May 20 19:24:36:60 RF_CLIENT: Signal Fail.
DEBUG (ACU) May 20 19:24:36:69 (0) io_handler.c : Invalid instrument type .
DEBUG (ACU) May 20 19:24:36:69 (0) io_handler.c: An error was detected in the HPIB parse.
STATUS (SCM) May 20 19:24:37:44 ScheduleMonitor: Schedule Updated.
DEBUG (AUT) May 20 19:24:45:60 (0) AUTHENT: Received msg id = 0 from pid# 7 (orig pid=7).
DEBUG (AUT) May 20 19:24:45:61 (0) AUTHENT: Login attempt for user goliath.
DEBUG (AUT) May 20 19:24:48:97 (0) AUTHENT: Received msg id = 1 from pid# 7 (orig pid=7).
DEBUG (AUT) May 20 19:24:48:97 (0) AUTHENT: Logout attempt for user goliath.
DEBUG (AUT) May 20 19:24:53:33 (0) AUTHENT: Received msg id = 0 from pid# 7 (orig pid=7).
DEBUG (AUT) May 20 19:24:53:33 (0) AUTHENT: Login attempt for user sschaire.
STATUS (UIF) May 20 19:24:53:37 Attempting to Upgrade to Control Access.
DEBUG (AUT) May 20 19:24:53:37 (0) AUTHENT: Received msg id = 2 from pid# 7 (orig pid=7).
DEBUG (AUT) May 20 19:24:53:37 (0) AUTHENT: Upgrade attempt for user sschaire.
STATUS (AUT) May 20 19:24:53:37 AUTHENT: Control Granted.
STATUS (UIF) May 20 19:24:53:42 Control upgrade granted

7.5 Concept of Operation

Files may be FTP'ed to or from the DSC at any time. In order to command the DSC to process a new schedule, the Remote Computer must login and gain control privilege. The DSC is a single user system allowing only one operator or remote computer to manipulate schedules.

The following terms are used below:

Status Privilege/Status mode - A mode of the user interface (i.e. Status Privilege is only granted privilege) where the operator may see status update on the top level screen, however, the operator is unable to progress to lower screens to view or update the schedule. This is also a mode of the Remote Computer (i.e. Status Privilege is only granted privilege) after initial login or after downgrade.

Control Privilege/Control Mode - A mode of the user interface (i.e. user interface granted Control Privilege) where the operator has full access to all screens, or a mode of remote control (i.e. Remote Computer granted Control Privilege), where the Remote Computer is permitted to command a new schedule update.

Environment Variables - Some of the DSC software is configured via environment variables. The DSC is delivered configured and it is possible to change this configuration without changing software. These Environment Variables are typically set by Scientific Atlanta personnel.

Self Downgrade - Self Downgrade is a downgrade request by the user that is in Control Mode. (e.g. DSC user interface or Remote Computer). The other type of downgrade request is an automatic downgrade request which occurs when another entity (e.g. DSC user interface or Remote Computer) tries to upgrade while not in control mode. In that case, an automatic downgrade request is sent to the entity in control.

7.5.1 Remote Computer Login

The Remote Computer shall login by submitting a login message packet to the network using the TCP/IP socket protocol. The format for the user data portion of this packet is given in section A3.3.1. The packet provides user name and password. The Remote Computer connection to the DSC will be in a status mode. The Remote Computer may request status packets from the DSC at this time.

7.5.2 Remote Computer Request for Control Upgrade

The Remote Computer shall request control upgrade (privilege) access by issuing the Control Upgrade Request Message packet. (See section A3.3.3) It will then wait for the control privilege response message packet, which will inform it of the outcome of the request. The Remote Computer will be in a control mode and may modify DSC system state by requesting the update of schedule information.

7.5.3 Contention for Control Privilege

If the DSC user has Control Privilege access and the Remote Computer requests Control Privilege access, a dialog box shall appear on the DSC screen for a specified time-out period. The dialog box shall support two buttons labeled: **GRANT** and **DENY.** Pressing GRANT will grant Control Privilege access to the requester, and downgrade the DSC to Status Privilege. Pressing DENY will deny Control Privilege access. If no operator response is received within the time-out period, then the request is GRANTED. Time-out periods of 0 (instantaneous) and "infinite" are permitted, and are specified via environment variable.

7.5.4 Self Downgrade

An entity with Control Privilege access may downgrade itself to Status Privilege access. The DSC will perform this by menu selection. The Remote Computer shall issue a Downgrade Request Message packet to Remote Control Interface.

7.5.5 Logoff

When an entity logs off, it loses both Control and Status Privilege access. The Remote Computer may log off at anytime by issuing a Logoff Message packet. The DSC local operator may log off at any time by menu selection.

7.6 TCP/IP Socket Communication

The Remote Computer communicates with the DSC via one TCP/IP bidirectional socket signal port . The DSC communicates with the Data Stripper via a TCP/IP socket port. The signal ports are defined in the file named:

(**UNIX**) → /etc/services

(**NT**) → c:\Winnt\system32\drivers\etc\services

Within the file the values for the ServiceName, Port#/Protocol fields are:

ServiceName	**Port#/Protocol**	**comment**
rci_recv	3015/tcp	# DSC signal port to receive commands from # Remote Computer
rci_send	3015/tcp	# DSC signal port to send responses, status, and # completion notification to Remote Computer
dsgems_async	4000/tcp	# DS signal port to send responses, status, and # completion notification to DSC

7.6.1 Remote Computer and DSC

The DSC is waiting on the rci_recv port, using bind, listen and accept commands. When a connection is established from the Remote Computer, the DSC uses the Remote Computer address from the rci_recv port to establish a socket address for sending using the rci_send port. This is achieved by using the remote socket address obtained during the accept. DSC and Data Stripper

The DSC is listening on the dsgems_async socket, using connect commands. When a connection is established with the Data Stripper the DSC will receive asynchronous communications for status, and event notification.

7.7 Internet Hosts File

The DSC communicates with the Data Stripper GEMS VIP Stations using RPC and Sockets. The TCP/IP addresss and hostnames are defined in the file named:

(**UNIX**) → /etc/hosts

(**NT**) → c:\Winnt\system32\drivers\etc\hosts

Within the file the values for the IP Address and hostname fields are:

Wallops Flight Facility

IP Address	Hostname	Hostname Alias	Comment
# DATA STRIPPER IP INFO			
#			
# rpc interface for synchronous communication of control			
ipaddress	dsgems1	stripper1fast	# primary vip station external interface
ipaddres	dsgems2	stripper3fast	# secondary vip station external interface
#			
# fastcopy, nfs, ftp interface for accessing lzp data			
ipaddres	dslzp1	stripper2fast	# primary lzp station external interface
ipaddres	dslzp2	stripper4fast	# secondary lzp station external interface
#			
# socket interface for asynchronous communication of status			
ipaddres	dsgems1int	stripper1	# primary vip station internal interface
ipaddres	dsgems2int	stripper3	# secondary vip station internal interface

Alaska SAR Facility

# DATA STRIPPER IP INFO			
#			
# rpc interface for synchronous communication of control			
ipaddres	dsgems1	stripper5fast	# primary vip station external interface
ipaddres	dsgems2	stripper7fast	# secondary vip station external interface
#			
# fastcopy, nfs, ftp interface for accessing lzp data			
ipaddres	dslzp1	stripper6fast	# primary lzp station external interface
ipaddres	dslzp2	stripper8fast	# secondary lzp station external interface
#			
# socket interface for asynchronous communication of status			
ipaddres	dsgems1int	stripper5	# primary vip station internal interface
ipaddres	dsgems2int	stripper7	# secondary vip station internal interface

APPENDIX B.

LV0P File Description

8.0 Appendix B - LV0P File Description

8.1 File Structure

LV0P is an individual use file prepared by MMO to inform the NASA and Kiruna stations of level 0 processing information of ADEOS-II mission data. MMO will prepare LV0P before 8:00 UTC on Monday, Wednesday, and Friday in the same schedule of orbital data (ELMP and ELMD) updates. In case that a due data is NASDA's holiday, LV0P preparation schedule updates follows that of the orbital data..

File Structure

Overseas LV0P Header Record
Data record 1: Level 0 Processing Information
Data record 2: Level 0 Processing Information 2
Data record 3: Level 0 Processing Information 3
.
.
.
.
.
Data record N: Level 0 Processing Information N

Note:

1. Each data record is sorted by the "Begin Date of Downlink Segment"

2. LV0P delivery and its data coverage are as follows:

Delivery	Coverage
Monday	Wednesday and Thursday Downlink (2 days)
Wednesday	Friday and Saturday Downlink (2 days)
Friday	Sunday, Monday and Tuesday Downlink (3 days)

3. See Table 3-2 for LV0P data coverage.

8.2 Header Record

No.	Field	Contents Description	Bytes	Byte #
1	File Name	LV0Pnnnnnn nnnnnn: file sequential number	10	0
2	blank	0x20 (delimiter)	1	10
3	Project Name	XXXXXX 'ADEOS2'(fixed)	6	11
4	blank	0x20 (delimiter)	1	17
5	EOC MMO Code (from)	XXXX 'HMMO' (fixed)	4	18
6	blank	0x20 (delimiter)	1	22
7	Overseas Station Code (to)	XXXX 'ASF *****': Alaska SAR Facility 'WFF *****': Wallops Flight Facility 'KRNS': Kiruna station	4	23
8	blank	0x20 (delimiter)	1	27
9	File Creation Date	YYYYMMDD date(UTC)	8	28
10	blank	0x20 (delimiter)	1	36
11	File Creation Time	hh:mm:ss time(UTC)	8	37
12	blank	0x20 (delimiter)	1	45
13	Length of Data Record	NNNN ' *****149'	4	46
14	blank	0x20 (delimiter)	1	50
15	Number of Data Records	NNNNN number of L0 processing plans	5	51
16	blank	0x20 (delimiter)	1	56
17	Begin Date of Data to be Processed (UTC)	YYYYMMDD Date of the acquisition start at the 1st record	8	57
18	blank	0x20 (delimiter)	1	65
19	End Date of Data to be Processed (UTC)	YYYYMMDD Date of the acquisition end at the last record	8	66
20	blank	0x20 (delimiter)	1	74
21	File Format Version (date)	YYYYMMDD date when the file format is authorized	8	75
22	blank	0x20 (delimiter)	1	83
23	File Format Version (number)	VNN NN: version number	3	84
24	blank	0x20 (delimiter)	1	87
25	Reserved	all blank space (0x20)	39	88
26	Record End	0x0A	1	127
		Sum	**128**	

8.3 Data Record

No.	Field	Contents Description	Byte	Byte #
1	Downlink Path Number	Pddddnnssss	11	0
2	blank	0x20 (delimiter)	1	11
3	Downlink Segment Number	Dxxxddddsss-zz	14	12
4	blank	0x20 (delimiter)	1	26
5	Start RSP of Downlink Segment	PPPPP *****AAA.AA RSP at the start of downlink segment	12	27
6	blank	0x20 (delimiter)	1	39
7	Stop RSP of Downlink Segment	PPPPP *****AAA.AA RSP at the stop of downlink segment	12	40
8	blank	0x20 (delimiter)	1	52
9	Time from A.N. for the Start of Downlink Segment(sec)	NNNNN Time is measured from the ascending node.	5	53
10	blank	0x20 (delimiter)	1	58
11	Time from A.N. for the Stop of Downlink Segment(sec)	NNNNN Time is measured from the ascending node.	5	59
12	blank	0x20 (delimiter)	1	64
13	Begin Date of Downlink Segment (UTC)	YYYYMMDD *****hh:mm:ss absolute time for the start of downlink seg.	17	65
14	blank	0x20 (delimiter)	1	82
15	End Date of Downlink Segment (UTC)	YYYYMMDD *****hh:mm:ss absolute time for the stop of downlink seg.	17	83
16	blank	0x20 (delimiter)	1	100
17	Acquisition Frequency Band	XXX 'X1 *****' = X1 band 'X3 *****' = X3 band	3	101
18	blank	0x20 (delimiter)	1	104
19	Acquisition Mode	XXX 'GLI' = real-time GLI250m data 'MDR' =MDR reproduction of multiple data 'ODM' =ODR reproduction of multiple data 'ODR' =ODR reproduction of GLI250m data 'MRT' =multiple real-time data capture	3	105
20	blank	0x20 (delimiter)	1	108
21	Station Code	XXXX 'ASF *****': Alaska SAR Facility 'WFF *****': Wallops Flight Facility 'KRNS': Kiruna station	4	109
22	blank	0x20 (delimiter)	1	113
23	Assigned IOCS	NNNNNN 'DIRECT'(fixed) IOCS name is set here especially for EOC.	6	114
24	blank	0x20 (delimiter)	1	120
25	Orbit Total Number for the Path	NNNNN	6	121

26	blank	0x20 (delimiter)	1	126
27	GLI 1km Processing Class	N '0': No Data ‘4’: raw data delivery	1	127
28	blank	0x20 (delimiter)	1	128
29	AMSR Processing Class	N ’0’: No Data ‘1’: only sensor data	1	129
30	blank	0x20 (delimiter)	1	130
31	DCS Processing Class	N ’0’: No Data ‘1’: only sensor data	1	131
32	blank	0x20 (delimiter)	1	132
33	SeaWinds Processing Class	N ’0’: No Data ‘1’: only sensor data	1	133
34	blank	0x20 (delimiter)	1	134
35	TEDA Processing Class	N ’0’: No Data ‘1’: only sensor data	1	135
36	blank	0x20 (delimiter)	1	136
37	ILAS-2 Processing Class	N ’0’: No Data ‘1’: only sensor data	1	137
38	blank	0x20 (delimiter)	1	138
39	POLDER Processing Class	N ’0’: No Data ‘1’: only sensor data	1	139
40	blank	0x20 (delimiter)	1	140
41	GLI 250m Processing Class	N '0': No Data '4': raw data delivery	1	141
42	blank	0x20 (delimiter)	1	142
43	VMS Processing Class	N ’0’: No Data ‘1’: only sensor data	1	143
44	blank	0x20 (delimiter)	1	144
45	DMS Processing Class (Reported back as two files, DMS-1 and DMS-2)	N ’0’: No Data ‘1’: only sensor data	1	145
46	blank	0x20 (delimiter)	1	146

47	Necessity of HKTLM Processing	N '0': not necessary '1': generate HKTLM source packet	1	147
48	Record End	0x0A	1	148
		Sum	**149**	

APPENDIX C.

RTIG File Description

9.0 Appendix C - RTIG File Description

9.1 File Structure

RTIG is an individual use file prepared by MMO to inform the NASA and Kiruna stations and NOAA of the ADEOS-II GLI 1km data acquisition plan based on the NOAA CoastWatch Program. MMO will prepare RTIG at 8:00 UTC on Monday, Wednesday, and Friday in the same schedule of orbital data (ELMP and ELMD) updates. In case that a due data is NASDA's holiday, RTIG preparation schedule updates follows that of the orbital data..

File Structure

RTIG Header Record
Data record 1: Selection Information
Data record 2: Selection Information 2
Data record 3: Selection Information 3
.
.
.
.
.
Data record N: Selection Information N

Note:

1. Each data records are divided by both the NOAA selected region and the acquisition segment.

2. RTIG delivery and its data coverage follow LV0P convention.

9.2 Header Record

No.	Field	Contents Description	Bytes	Byte #
1	File Name	RTIGnnnnnn nnnnnn: file sequential number	10	0
2	blank	0x20 (delimiter)	1	10
3	Project Name	XXXXXX 'ADEOS2'(fixed)	6	11
4	blank	0x20 (delimiter)	1	17
5	EOC MMO Code (from)	XXXX 'HMMO' (fixed)	4	18
6	blank	0x20 (delimiter)	1	22
7	Overseas Station Code (to)	XXXX 'ASF *****': Alaska SAR Facility 'WFF *****': Wallops Flight Facility 'NOAA': NOAA/NESDIS 'KRNS': Kiruna Station	4	23
8	blank	0x20 (delimiter)	1	27
9	File Creation Date	YYYYMMDD date(UTC)	8	28
10	blank	0x20 (delimiter)	1	36
11	File Creation Time	hh:mm:ss time(UTC)	8	37
12	blank	0x20 (delimiter)	1	45
13	Length of Data Record	NNNN ' *****168': (fixed)	4	46
14	blank	0x20 (delimiter)	1	50
15	Number of Data Records	NNNNN	5	51
16	blank	0x20 (delimiter)	1	56
17	Begin Date of Request	YYYYMMDD date(UTC)	8	57
18	blank	0x20 (delimiter)	1	65
19	End Date of Request	YYYYMMDD date(UTC)	8	66
20	blank	0x20 (delimiter)	1	74
21	File Format Version (date)	YYYYMMDD date when the file format is authorized	8	75
22	blank	0x20 (delimiter)	1	83
23	File Format Version (number)	VNN NN: version number	3	84
24	blank	0x20 (delimiter)	1	87
25	Reserved	all blank space (0x20)	39	88
26	Record End	0x0A	1	127
		Sum	**128**	

9.3 Data Record

No.	Field	Contents Description	Bytes	Byte#
1	Operation Plan Number	NGL1ddddsss	11	0
2	blank	0x20 (delimiter)	1	11
3	Acquisition Segment Number	SGL1ddddsss	11	12
4	blank	0x20 (delimiter)	1	23
5	Downlink Path Number	Pddddnnssss	11	24
6	blank	0x20 (delimiter)	1	35
7	Downlink Segment Number	Dxxxddddsss-zz	14	36
8	blank	0x20 (delimiter)	1	50
9	Start RSP of the Selected Data	PPPPP *****AAA.AA	12	51
10	blank	0x20 (delimiter)	1	63
11	Stop RSP of the Selected Data	PPPPP *****AAA.AA	12	64
12	blank	0x20 (delimiter)	1	76
13	Begin Date of the Selected Data	YYYYMMDD *****hh:mm:ss	17	77
14	blank	0x20 (delimiter)	1	94
15	End Date of the Selected Data	YYYYMMDD *****hh:mm:ss	17	95
16	blank	0x20 (delimiter)	1	112
17	Spacecraft Time Counter(start)	NNNNNNNNNNN sc time counter extrapolated from time difference data (see note for the definition)	11	113
18	blank	0x20 (delimiter)	1	124
19	Spacecraft Time Counter(stop)	NNNNNNNNNNN sc time counter extrapolated from time difference data (see note for the definition)	11	125
20	blank	0x20 (delimiter)	1	136
21	Time from the A.N. for the Start RSP (sec)	NNNNN Time is measured from the ascending node.	5	137
22	blank	0x20 (delimiter)	1	142
23	Time from the A.N. for the Stop RSP (sec)	NNNNN Time is measured from the ascending node.	5	143
24	blank	0x20 (delimiter)	1	148
25	ID of the Selected Data	RTIG-PP-NNNNN PP: Path number of Start RSP (field #9) NNNNN: seconds from A.N. (field #21)	13	149
26	blank	0x20 (delimiter)	1	162
27	User Agency Code	NNNN 'NOAA'(fixed)	4	163
28	record end	0x0A	1	167
		Sum	**168**	

Note:

1. "ID of the Selected Data" is assigned when 4-week assessment is completed. This number is the unique key to identify each selected data-sets.
2. NOAA Selected Regions.
   (1) The NOAA selected regions that are observed in descending orbits are informed by this file.
   (2) Short gaps between given two regions are ignored, then continuous RSP are set instead of divided small ranges. Threshold of this case is TBD.
3. JPL AMSR Request
   TBD
4. JAFIC AMSR Request
   TBD
5. The definition of 'Space Craft Time Counter' is the same as 'Reference Counter of SC Clock' in TMDF file.
   It's described in the unit of second.

APPENDIX D.

L0RL File Description

10.0 Appendix D - L0RL File Description

10.1 File Structure

L0RL is an individual use file prepared by the NASA and Kiruna stations to inform MMO of level 0 processing results of ADEOS-II mission data in response to LV0P. The NASA and Kiruna stations place LV0P in the designated system after each L0 processing for EOC retrieval.

One L0RL file covers one downlink segment including a corresponding set of the acquisition segment.

File Structure

Overseas L0RL Header Record
Data record 1: Level 0 Processing Result
Data record 2: Level 0 Processing Result 2
Data record 3: Level 0 Processing Result 3
.
.
.
.
.
Data record N: Level 0 Processing Result N

Notes:

1. When a level 0 product is sent to two agencies simultaneously, two data records must be set in one file. The number of agencies equals to that of data records.

10.2 Header Record

No.	Field	Contents Description	Bytes	Byte #
1	File Name	L0RLnnnnnn nnnnnn: file sequential number	10	0
2	blank	0x20 (delimiter)	1	10
3	Project Name	XXXXXX 'ADEOS2'(fixed)	6	11
4	blank	0x20 (delimiter)	1	17
5	Overseas Station Code (to)	XXXX 'ASF *****': Alaska SAR Facility 'WFF *****': Wallops Flight Facility 'KRNS': Kiruna station	4	18
6	blank	0x20 (delimiter)	1	22
7	EOC MMO Code (from)	XXXX 'HMMO' (fixed)	4	23
8	blank	0x20 (delimiter)	1	27
9	File Creation Date	YYYYMMDD date(UTC)	8	28
10	blank	0x20 (delimiter)	1	36
11	File Creation Time	hh:mm:ss time(UTC)	8	37
12	blank	0x20 (delimiter)	1	45
13	Length of Data Record	NNNN ' *****203' (fixed)	4	46
14	blank	0x20 (delimiter)	1	50
15	Number of Data Records	NNNNN number of downlink segments	5	51
16	blank	0x20 (delimiter)	1	56
17	Begin Date of Data	YYYYMMDD begin date of acquisition of the 1st record(UTC)	8	57
18	blank	0x20 (delimiter)	1	65
19	End Date of Data	YYYYMMDD end date of acquisition of the last record(UTC)	8	66
20	blank	0x20 (delimiter)	1	74
21	File Format Version (date)	YYYYMMDD date when the file format is authorized	8	75
22	blank	0x20 (delimiter)	1	83
23	File Format Version (number)	VNN NN: version number	3	84
24	blank	0x20 (delimiter)	1	87
25	Reserved	all blank space (0x20)	39	88
26	Record End	0x0A	1	127
		Sum	**128**	

10.3 Data Record

No.	Field	Contents Description	Bytes	Byte #
1	Downlink Path Number	Pddddnnssss	11	0
2	blank	0x20 (delimiter)	1	11
3	Downlink Segment Number	Dxxxddddsss-zz	14	12
4	blank	0x20 (delimiter)	1	26
5	Station Code(X-band)	NNNN 'ASF *****': Alaska SAR Facility 'WFF *****': Wallops Flight Facility 'KRNS': Kiruna station	4	27
6	blank	0x20 (delimiter)	1	31
7	Processing Information	NNNN 'RECV' (fixed)	4	32
8	blank	0x20 (delimiter)	1	36
9	Classifier of L0 Data	N '1': only sensor data (fixed)	1	37
10	blank	0x20 (delimiter)	1	38
11	Sensor Name	NNN	3	39
12	blank	0x20 (delimiter)	1	42
13	Destination Agency/Station	NNNN 'HEOC' (fixed)	4	43
14	blank	0x20 (delimiter)	1	47
15	Input of L0 Processing	NNNN 'REAL' (fixed)	4	48
16	blank	0x20 (delimiter)	1	52
17	Date of RAW Data Received	YYYYMMDD *****hh:mm:ss '******** *********' (fixed)	17	53
18	blank	0x20 (delimiter)	1	70
19	File Name of L0 Product	NNNNNN… *** * *** file name of L0 product sent via network	39	71
20	blank	0x20 (delimiter)	1	110
21	Date of L0 Processing Completion	YYYYMMDD *****hh:mm:ss	17	111
22	blank	0x20 (delimiter)	1	128
23	L0 Processing Result	NNNN 'OK *** ***': normally completed 'NG *** ***': abnormal end	4	129
24	blank	0x20 (delimiter)	1	133
25	Date of L0 Delivery	YYYYMMDD *****hh:mm:ss	17	134
26	blank	0x20 (delimiter)	1	151
27	Status of L0 Delivery	NNNNN 'OK *** * ***': normally completed 'NG *** * ***': abnormal end	5	152
28	blank	0x20 (delimiter)	1	157
29	Lost Packets	NNNNNNNN	8	158
30	blank	0x20 (delimiter)	1	166

31	Total Packets	NNNNNNNN	8	167
32	blank	0x20 (delimiter)	1	175
33	Lost Major Frames	NNNNNNNN '********' (fixed)	8	176
34	blank	0x20 (delimiter)	1	184
35	Lost Minor Frames	NNNNNNNN '********' (fixed)	8	185
36	blank	0x20 (delimiter)	1	193
37	Total Minor Frames	NNNNNNNN '********' (fixed)	8	194
38	Record End	0x0A	1	202
		Sum	**203**	

APPENDIX E.

DSC Status Log File Description

11.0 Appendix E - DSC Status Log File Description

11.1 File Structure

The DSC Status Log file is an ASCII text-format file containing keyword/value information about the MDR and MRT data streams of the associated support. Whitespace characters (consecutive blanks and tabs, in any order) separate the keyword with its string or numeric value. A colon character immediately follows the keyword and a New Line character terminates each line. Blank lines are possible.

The following shows an example of a DSC Status Log file. The "[X1 stream]" identifier gives the statistics for the MDR data stream used to generate the output Level 0 files. The "[X3 stream]" identifier gives the statistics for the MRT data stream. The values for the frame count fields and the "Back to search" field are integer counts, while the "Bit error rate" value is in units of errors per 10^9 bits.

Unit: WFF/ASF Data Stripper Controller
File date: 174 20:44:34 Jan 23, 2001
Filename: ftp://*strpboss*/*export/home/dscs/etc*/log/reports/ADEOS2.039.13.48

[X1 stream]
Total frames: 1247964
Corrected frames: 139
Uncorrected frames: 73
Back to search: 4
Bit error rate: 14

[X3 stream]
Total frames: 174027
Corrected frames: 35
Uncorrected frames: 14
Back to search: 3
Bit error rate: 19

APPENDIX F.

Level Zero Data File Naming Convention

12.0 Appendix F – Level Zero Data File Naming Convention

12.1 File Names

Level 0 files are sent to SAFS using FastCopy. The files are sent to a specific directory path on SAFS depending on the file type as given by the Sensor Name shown in the table below:

Sensor	Type	SAFS Directory	Name field: "sen"
AMSR	ams	/raid1/safs/ad2/ams/	AMS
DCS	dcs	/raid1/safs/ad2/dcs/	DCS
DMS1	dms	/raid1/safs/ad2/dms/	DM1
DMS2	dms	/raid1/safs/ad2/dms/	DM2
GLI-1K	gli	/raid1/safs/ad2/gli/	GL1
HK	hkm	/raid1/safs/ad2/hkm/	HKT
ILAS	ila	/raid1/safs/ad2/ila/	ILA
SeaWinds	swm	/raid1/safs/ad2/swm/	SEA
TEDA	ted	/raid1/safs/ad2/ted/	TED
VMS	vms	/raid1/safs/ad2/vms/	VMS

The files (data and status) sent to SAFS have the following name format:

projectFileName=safsFileName=

or, expanded into its component fields:

si_sen_amd_ffid_dt_dst_yyyymmdd_snnn=pppyyyymmddhhmmssdnnttt.ext=

where the "project name" and "safs name" fields are given in the tables below. Note that fields in the "Project Name" are in upper case letters and are separated by underscore ("_") characters in the file name while fields in the "SAFS Name" are concatenated with no separator characters, and are in lower case.

Field	Description	Values
si	Satellite I.D.	fixed "A2"
sen	Sensor name (see Name Field: "sen" above)	ex. "AMS", "DM1"
amd	Acquisition mode	"MDR" or "MRT"
ffid	File Generation Facility I.D.	"ASF-" or "WFF-"
dt	Data Type	fixed "L0"
dst	Data SubType (L0 data "SIG" or Status File "STR")	"SIG" or "STR"
yyyymmdd	Date (year/month/day) of file generation	ex. "20010328"
snnn	L0 Processor I.D. "1" and L0 file sequence in day	ex. "1013"
ppp	Project name	Fixed "ad2"
yyyymmdd	Date (year/month/day) of pass recording start time	ex. "20010328"
hhmmss	Time (hour/min/sec) of pass recording start time	ex. "154207"
dnn	TM Processor I.D. ("d01")	Fixed. "d01"
ttt	Sensor type (see "Type" field in table above)	ex. "ila", "ted"
ext	File extension (L0 data "dat" or Status File ("mta")	"dat" or "mta"